

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

THIS PAGE BLANK (USPTO)



XP 000259620

Entwicklungssysteme

Emulatoren

E

p. 100 - 103

(h)

2087 Elektronik
40(1991)6 August, No. 16, München, DE

In zunehmendem Maße werden Mikrocontroller in Anwendungen eingesetzt, die sehr komplex und zeitkritisch sind und dazu noch ein hohes Maß an Zuverlässigkeit und Fehlertoleranz verlangen, so daß die Anforderungen an Entwicklungs- und Testgeräte – insbesondere an die In-Circuit-Emulatoren – immer höher geschraubt werden. Doch können die Emulatoren auch im Echtzeitbereich noch mit halten?

GIB 15/46

Dipl.-Ing. Stefan Richt, Peter Hamm

In-Circuit-Emulator wird „echtzeittauglich“

Praxis-Beispiel aus der Regelungstechnik

Dieser Beitrag soll anhand eines Projektes die Eigenschaften beschreiben, die für Embedded-Controller-Entwicklungssysteme heute sehr wünschenswert sind. Beispielhaft herangezogen wird hierzu eine selbst konzipierte Entwicklungsumgebung.

Aufgabenstellung: Design eines Steuerrechners

Es sollte ein Steuerrechner mit einem 8032-Prozessor für ein analoges Kassettengerät entwickelt werden, der die Überwachung des Bandlaufes, die serielle Kommunikation und die Regelung der Bandgeschwindigkeit simultan für mehrere Kassettenlaufwerke übernehmen sollte. Als Programmiersprache war C vorgesehen.

Die Anforderungen an die Bandgeschwindigkeits-Regelung wurden sehr hoch gesteckt: Kurzzeitige Geschwindigkeitsabweichungen (Gleichlauf) durften maximal 0,07 % betragen, und die Langzeitkonstanz (Drift) sollte besser als 0,005 % sein. Die Bandlaufüberwachung mußte Fehlerzustände wie „Bandklemmen“ und „Bandschläufe“ erkennen und verhindern, daß das Band beschädigt wird. Eine serielle Kommunikation mit einem externen Rechner sollte implementiert werden.

Die verwendeten Tools

Eingesetzt wurde ein selbstentwickelter In-Circuit-Emulator und ein ebenfalls selbstentwickelter Hardware-Simulator. In diesen Tools ließen sich Wünsche realisieren, die sich in mehreren Jahren bei der Entwicklung von „Embedded-Control“-Anwendungen herauskristallisiert hatten. In der Codierphase wurde hauptsächlich mit dem Simulator gearbeitet.

Die Implementierung der Firmware im Controller und das Testen bzw. Optimieren der Regelungs- und

Überwachungsfunktionen erfolgte mit dem In-Circuit-Emulator.

Ohne „Interrupts“ geht es nicht

Es zeigte sich, daß alle zeitkritischen Aufgaben in Interruptroutinen untergebracht werden mußten. Die Messung der Bandgeschwindigkeit wurde wegen der geforderten Meßgenauigkeit mit Hilfe der Capture-Register von Timer 2 vorgenommen. Die Kommunikation, die Überwachung des Bandlaufes sowie der Regelalgorithmus mußten als Interruptroutinen realisiert werden; lediglich der Interpreter, der die Kommandos der seriellen Schnittstelle bearbeitete, und die Ansteuerung der Stellglieder konnten im Hauptprogramm untergebracht werden. Die Regelung selbst wurde als Kaskadenregelung mit einer analogen Frequenzvorregelung realisiert, da der digitale Regelalgorithmus zu langsam war, um den geforderten Gleichlauf zu ermöglichen. Die Regelung wurde schließlich verschachtelt und die Zeitmeßeingänge und die Stellausgänge gemultiplext, um die Ansteuerung mehrerer Laufwerke zu ermöglichen.

Im Verlauf der Projekts stellte sich heraus, daß die Implementierung der Regelung der anspruchsvollste Teil der Aufgabe war.

Was macht das Testen einer digitalen Regelung schwierig?

Damit eine Regelung funktioniert, müssen mehrere Dinge gleichzeitig sichergestellt werden:

- Der Regelalgorithmus muß fehlerfrei sein.
- Der Fehler bei der Messung der Istgröße muß geringer sein als die zulässige Regelabweichung.

- Die Regelparameter müssen abgestimmt sein.
- Der Regelkreis muß geschlossen sein.

Module, die zur Regelung gehören, können zwar einzeln vorgetestet werden, dennoch läßt sich das dynamische Verhalten einer Regelung nur in Echtzeit beurteilen. Gerade hier sind leistungsfähige Entwicklungswerkzeuge gefragt, die über erweiterte „Echtzeit“-Testmöglichkeiten verfügen.

Die Interruptroutinen: aktiv im Hintergrund

Der Steuerrechner sollte neben der Regelung der Bandgeschwindigkeit auch die Kontrolle der seriellen Schnittstelle und die Überwachung des Bandlaufes erledigen. Der Bandlauf läßt sich aber erst dann sinnvoll überwachen, wenn die Bandgeschwindigkeit korrekt eingestellt ist: Deshalb mußten die Interruptroutinen der Regelung zuerst implementiert werden und während des gesamten Systemtests im Hintergrund aktiv bleiben.

Mit einem Emulator, der die Abarbeitung von Interruptroutinen bei stehender Emulation nicht unterstützt, hätte man diese Anwendung nur noch mit dem Trace (Verfolgungsspeicher) testen können, nachdem die erste Interruptroutine implementiert worden war. Dies wäre jedoch eine sehr unkomfortable Lösung.

Da der Mikrocontroller-Emulator auch die Abarbeitung von Interruptroutinen im Hintergrund unterstützen sollte, wurde eine Interrupt-Erkennung und Prioritäten-Zuordnung implementiert, die universell mit allen 8051-Derivaten zusammenarbeitet, bis zu vier Prioritätsebenen berücksichtigt und vom Anwender keinerlei Eingaben erfordert.

Dadurch ist es möglich, das Programm wie gewohnt zu „testen“, d.h. alle zur Verfügung stehenden Debug-Befehle wie „Single-Step“, „Snapshot“ oder „Breakpoint“ auszuführen, während im Hintergrund alle aktiven Interruptroutinen in Echtzeit abgearbeitet werden.

Eine ungewöhnliche Programmtest-Strategie

Daraus ergibt sich eine für viele Emulator-Besitzer ungewohnte, da von ihren Geräten nicht unterstützte, aber sehr effiziente und bequeme Testmethode: Man implementiert und prüft die Interruptroutine mit der höchsten Priorität zuerst. Ist diese „wichtigste“ Interruptroutine getestet, kommen die Interruptroutinen niedrigerer Priorität an die Reihe, wobei die „wichtigste“ Interruptroutine weiterhin in Echtzeit läuft. Sind alle Interruptroutinen getestet und fehlerfrei, kann zuletzt bequem das Hauptprogramm geprüft und optimiert werden, wobei alle Interruptroutinen weiterhin in Echtzeit abgearbeitet werden.

Interaktive Abstimmung der Regelparameter bei laufender Regelung

Während die Regelung in Echtzeit lief, konnten mit dem Emulator interaktiv die Parameter geändert werden. Da die Wirkungen unmittelbar sichtbar waren, ließ

sich die Regelung bequem abstimmen. Es wurde ein PID-Regler eingesetzt; zunächst als reiner P- (Proportional-)Regler implementiert. Der Proportional-Anteil wurde anschließend so lange verändert, bis das System an die Stabilitätsgrenze kam. Die Einstellung der I- (Integral-), D- (Differential-) und P- Anteile erfolgte zunächst grob nach Einstellregeln und konnte interaktiv schnell am System optimiert werden.

Ohne einen leistungsfähigen Analysator geht es nicht

Es gibt Situationen, in denen ein Entwickler über einen leistungsfähigen Trace-Analysator verfügen sollte, denn komplizierte Soft- oder Hardwarefehler lassen sich meist nur damit aufspüren.

Was heißt „unscharf triggern“?

Angenommen sei das Beispiel, in dem sich ein Fehler dadurch zeigt, daß eine Float-Variable einen falschen Wert annimmt. Hier ist nun nicht auf den Wert der Variablen selbst zu triggern, sondern es genügt, auf die Adresse der Variablen zu triggern, was mit jeder Breakpoint-Logik einfach möglich sein sollte. In den Trace-Speicher werden alle Speicherzugriffe auf die Variable inklusive Pre-Trace geschrieben. Der Trace-Speicher läßt sich nun mit oder ohne Software-Unterstützung analysieren und somit die fehlerhafte Variable suchen. In der zusätzlich zu der Variablen aufgezeichneten Vorgeschichte (Pre-Trace) kann die Ursache des Fehlers lokalisiert werden.

Die meisten Emulatoren verfügen als Zusatzoption über einen solchen Analysator. Abhängig von einem Triggersignal werden dabei Bus-Zyklen und Daten von Probes in einen Speicher geschrieben, dessen Inhalt man zur Programmanalyse wieder auslesen kann. Bekannte Aufzeichnungsarten sind Pre-, Post- und Center-Trace-Modus. Im Pre-Trace-Modus ist die Aufzeichnung zunächst aktiv, mit dem Triggersignal wird die Aufzeichnung gestoppt. Im Speicher stehen die Daten zum Zeitpunkt des Triggersignals (Triggerereignis) und Daten, die zeitlich vor dem Triggersignal lagen (Pre-Trace). Im Post-Trace-Modus wird die Aufzeichnung durch das Triggersignal gestartet und bei einer bestimmten Abbruchbedingung (z.B. nach 100 Zyklen) wieder gestoppt. Im Speicher stehen das Triggerereignis selbst und Daten, die nach dem Triggersignal aufgetreten sind (Post-Trace). Der Center-Trace-Modus stellt eine Kombination aus beiden Modi dar. Das Triggersignal wird von einer Breakpoint-Logik erzeugt, die der

Entwicklungsstysteme

Emulatoren

Bild 1. Multiple-Center-Trace-Aufzeichnung auf Source-Code-Ebene: aktives Source-Line-Filter, Pre-Trace mit zwei Zyklen, Post-Trace mit drei Zyklen.

Entwickler programmieren kann, um die Aufzeichnung zu steuern.

Ist „Single Pre-Trace“-Modus ausreichend?

Da die Ursachen für Fehlerzustände zeitlich vor dem Auftreten des Fehlerzustandes selbst liegen, ist der Pre-Trace-Aufzeichnungsmodus meist das wirkungsvollste Mittel zur Fehlersuche. Nachteilig bei dem bekannten Pre-Trace-Modus ist die Tatsache, daß technisch bedingt nur ein Ereignis mit Pre-Trace im Speicher stehen kann.

Probleme treten dann auf, wenn auf die Auswirkung eines unbekannten Fehlers nicht getriggert werden kann. Bei der implementierten Regelung mußte man beispielsweise die Ursache für sporadisch auftretende, zu hohe Gleichlaufschwankungen finden. Wie kann man aber auf derartige „Ausreißer“ triggern, ohne das

```
Display Full Load Memory Execute Init Breakpoint Register Watch Help
analysator running
CHIP: E031 emulation running
status exec probe9 probe1

ANALYSATOR: -2000

c=c*c;
    pre 111.65 .....
    post 111.77 .....
    post 111.78 .....

while (i < 100) {
    b=sqrta();
    b=b*2*a/7;
    c=c+b+a/8.0;
    c=c*c;
    i++;
    i++;
}

while (i < 100) {
    b=sqrta();
    b=b*2*a/7;
    c=c+b+a/8.0;
    c=c*c;
    i++;
    i++;
}

pre 115.55 .....
pre 115.57 .....
pre 117.64 .....
event 118.71 .....
post 119.82 .....
post 119.14 .....
post 119.14 .....
```

Gleichlauf-Meßgerät umzubauen und mit einem Trigerausgang zu versehen? Ein anderes Beispiel kann ein Fehler sein, bei dem eine Float-Variable einen unerlaubten Wert annimmt.

Selbst mit einer äußerst leistungsfähigen sequentiellen Breakpoint-Logik kann in Echtzeit kaum auf den Wert einer Float-Variablen getriggert werden. – Doch es gibt einen Ausweg.

„Multiple Center-Trace“ ist die Lösung

Mit Hilfe des „Multiple Center-Trace“-Modus ist es möglich, auch „nichttriggerbare Fehler“ im Trace-Speicher aufzuzeichnen und somit zu finden. Doch was bedeutet „Multiple Center-Trace“-Modus?

Zusätzlich zu jedem Triggerereignis können ein Pre-Trace und ein Post-Trace aufgezeichnet werden. Die

Länge von Pre- und Post-Trace ist getrennt einstellbar und kann zwischen 0 und 512 Zyklen liegen (Bild 1). Ein Speichereintrag, der aus Triggerereignis, Pre- und Post-Zyklen besteht, läßt sich dann treffend als „Frame“ bezeichnen. In Echtzeit und ohne Unterbrechung der Trace-Aufzeichnung können beliebig viele „Frames“ in den Trace-Speicher aufgezeichnet werden. Die Anzahl der „Frames“ hängt von der eingestellten Länge des Pre- und des Post-Trace sowie von der Tiefe des Trace-Speichers ab. Im 32 KWorte tiefen Trace-Speicher des Emulators lassen sich z. B. 16 KFrames mit jeweils einem Pre-Trace-Zyklus aufzeichnen.

Durch den „Multiple Center Trace“-Modus ist man nun bei der Fehlersuche in der Lage, „unscharf“ zu triggern, das heißt, es ist nicht notwendig, auf die exakte Fehlerbedingung selbst zu triggern, sondern es genügt, die Triggerbedingung so zu definieren (Bild 2), daß wenigstens einer der aufgezeichneten Frames den Fehler enthält.

```
Display Full Load Memory Execute Init Breakpoint Register Watch Help
analysator running
CHIP: E031 emulation running
status exec probe9 probe1

ANALYSATOR: -2000

test1 test1
    pre 2.015 .....
    post 2.016 .....
    post 2.017 .....
    post 2.018 .....

test2 test2
    pre 3.015 .....
    pre 3.016 .....
    event 3.017 .....
    post 3.018 .....
    post 3.019 .....
    post 3.020 .....

test3 test3
    pre 3.415 .....
    pre 3.416 .....
    event 3.417 .....
    post 3.418 .....
    post 3.419 .....
    post 3.420 .....

test4 test4
    pre 4.015 .....
    pre 4.016 .....
    event 4.017 .....
    post 4.018 .....
    post 4.019 .....
    post 4.020 .....

test5 test5
    pre 4.415 .....
    pre 4.416 .....
    event 4.417 .....
    post 4.418 .....
    post 4.419 .....
    post 4.420 .....

test6 test6
    pre 5.015 .....
    pre 5.016 .....
    event 5.017 .....
    post 5.018 .....
    post 5.019 .....
    post 5.020 .....

test7 test7
    pre 5.415 .....
    pre 5.416 .....
    event 5.417 .....
    post 5.418 .....
    post 5.419 .....
    post 5.420 .....

test8 test8
    pre 6.015 .....
    pre 6.016 .....
    event 6.017 .....
    post 6.018 .....
    post 6.019 .....
    post 6.020 .....

test9 test9
    pre 6.415 .....
    pre 6.416 .....
    event 6.417 .....
    post 6.418 .....
    post 6.419 .....
    post 6.420 .....

test10 test10
    pre 7.015 .....
    pre 7.016 .....
    event 7.017 .....
    post 7.018 .....
    post 7.019 .....
    post 7.020 .....

test11 test11
    pre 7.415 .....
    pre 7.416 .....
    event 7.417 .....
    post 7.418 .....
    post 7.419 .....
    post 7.420 .....

test12 test12
    pre 8.015 .....
    pre 8.016 .....
    event 8.017 .....
    post 8.018 .....
    post 8.019 .....
    post 8.020 .....

test13 test13
    pre 8.415 .....
    pre 8.416 .....
    event 8.417 .....
    post 8.418 .....
    post 8.419 .....
    post 8.420 .....

test14 test14
    pre 9.015 .....
    pre 9.016 .....
    event 9.017 .....
    post 9.018 .....
    post 9.019 .....
    post 9.020 .....

test15 test15
    pre 9.415 .....
    pre 9.416 .....
    event 9.417 .....
    post 9.418 .....
    post 9.419 .....
    post 9.420 .....

test16 test16
    pre 10.015 .....
    pre 10.016 .....
    event 10.017 .....
    post 10.018 .....
    post 10.019 .....
    post 10.020 .....

test17 test17
    pre 10.415 .....
    pre 10.416 .....
    event 10.417 .....
    post 10.418 .....
    post 10.419 .....
    post 10.420 .....

test18 test18
    pre 11.015 .....
    pre 11.016 .....
    event 11.017 .....
    post 11.018 .....
    post 11.019 .....
    post 11.020 .....

test19 test19
    pre 11.415 .....
    pre 11.416 .....
    event 11.417 .....
    post 11.418 .....
    post 11.419 .....
    post 11.420 .....

test20 test20
    pre 12.015 .....
    pre 12.016 .....
    event 12.017 .....
    post 12.018 .....
    post 12.019 .....
    post 12.020 .....

test21 test21
    pre 12.415 .....
    pre 12.416 .....
    event 12.417 .....
    post 12.418 .....
    post 12.419 .....
    post 12.420 .....

test22 test22
    pre 13.015 .....
    pre 13.016 .....
    event 13.017 .....
    post 13.018 .....
    post 13.019 .....
    post 13.020 .....

test23 test23
    pre 13.415 .....
    pre 13.416 .....
    event 13.417 .....
    post 13.418 .....
    post 13.419 .....
    post 13.420 .....

test24 test24
    pre 14.015 .....
    pre 14.016 .....
    event 14.017 .....
    post 14.018 .....
    post 14.019 .....
    post 14.020 .....

test25 test25
    pre 14.415 .....
    pre 14.416 .....
    event 14.417 .....
    post 14.418 .....
    post 14.419 .....
    post 14.420 .....

test26 test26
    pre 15.015 .....
    pre 15.016 .....
    event 15.017 .....
    post 15.018 .....
    post 15.019 .....
    post 15.020 .....

test27 test27
    pre 15.415 .....
    pre 15.416 .....
    event 15.417 .....
    post 15.418 .....
    post 15.419 .....
    post 15.420 .....

test28 test28
    pre 16.015 .....
    pre 16.016 .....
    event 16.017 .....
    post 16.018 .....
    post 16.019 .....
    post 16.020 .....

test29 test29
    pre 16.415 .....
    pre 16.416 .....
    event 16.417 .....
    post 16.418 .....
    post 16.419 .....
    post 16.420 .....

test30 test30
    pre 17.015 .....
    pre 17.016 .....
    event 17.017 .....
    post 17.018 .....
    post 17.019 .....
    post 17.020 .....

test31 test31
    pre 17.415 .....
    pre 17.416 .....
    event 17.417 .....
    post 17.418 .....
    post 17.419 .....
    post 17.420 .....

test32 test32
    pre 18.015 .....
    pre 18.016 .....
    event 18.017 .....
    post 18.018 .....
    post 18.019 .....
    post 18.020 .....

test33 test33
    pre 18.415 .....
    pre 18.416 .....
    event 18.417 .....
    post 18.418 .....
    post 18.419 .....
    post 18.420 .....

test34 test34
    pre 19.015 .....
    pre 19.016 .....
    event 19.017 .....
    post 19.018 .....
    post 19.019 .....
    post 19.020 .....

test35 test35
    pre 19.415 .....
    pre 19.416 .....
    event 19.417 .....
    post 19.418 .....
    post 19.419 .....
    post 19.420 .....

test36 test36
    pre 20.015 .....
    pre 20.016 .....
    event 20.017 .....
    post 20.018 .....
    post 20.019 .....
    post 20.020 .....

test37 test37
    pre 20.415 .....
    pre 20.416 .....
    event 20.417 .....
    post 20.418 .....
    post 20.419 .....
    post 20.420 .....

test38 test38
    pre 21.015 .....
    pre 21.016 .....
    event 21.017 .....
    post 21.018 .....
    post 21.019 .....
    post 21.020 .....

test39 test39
    pre 21.415 .....
    pre 21.416 .....
    event 21.417 .....
    post 21.418 .....
    post 21.419 .....
    post 21.420 .....

test40 test40
    pre 22.015 .....
    pre 22.016 .....
    event 22.017 .....
    post 22.018 .....
    post 22.019 .....
    post 22.020 .....

test41 test41
    pre 22.415 .....
    pre 22.416 .....
    event 22.417 .....
    post 22.418 .....
    post 22.419 .....
    post 22.420 .....

test42 test42
    pre 23.015 .....
    pre 23.016 .....
    event 23.017 .....
    post 23.018 .....
    post 23.019 .....
    post 23.020 .....

test43 test43
    pre 23.415 .....
    pre 23.416 .....
    event 23.417 .....
    post 23.418 .....
    post 23.419 .....
    post 23.420 .....

test44 test44
    pre 24.015 .....
    pre 24.016 .....
    event 24.017 .....
    post 24.018 .....
    post 24.019 .....
    post 24.020 .....

test45 test45
    pre 24.415 .....
    pre 24.416 .....
    event 24.417 .....
    post 24.418 .....
    post 24.419 .....
    post 24.420 .....

test46 test46
    pre 25.015 .....
    pre 25.016 .....
    event 25.017 .....
    post 25.018 .....
    post 25.019 .....
    post 25.020 .....

test47 test47
    pre 25.415 .....
    pre 25.416 .....
    event 25.417 .....
    post 25.418 .....
    post 25.419 .....
    post 25.420 .....

test48 test48
    pre 26.015 .....
    pre 26.016 .....
    event 26.017 .....
    post 26.018 .....
    post 26.019 .....
    post 26.020 .....

test49 test49
    pre 26.415 .....
    pre 26.416 .....
    event 26.417 .....
    post 26.418 .....
    post 26.419 .....
    post 26.420 .....

test50 test50
    pre 27.015 .....
    pre 27.016 .....
    event 27.017 .....
    post 27.018 .....
    post 27.019 .....
    post 27.020 .....

test51 test51
    pre 27.415 .....
    pre 27.416 .....
    event 27.417 .....
    post 27.418 .....
    post 27.419 .....
    post 27.420 .....

test52 test52
    pre 28.015 .....
    pre 28.016 .....
    event 28.017 .....
    post 28.018 .....
    post 28.019 .....
    post 28.020 .....

test53 test53
    pre 28.415 .....
    pre 28.416 .....
    event 28.417 .....
    post 28.418 .....
    post 28.419 .....
    post 28.420 .....

test54 test54
    pre 29.015 .....
    pre 29.016 .....
    event 29.017 .....
    post 29.018 .....
    post 29.019 .....
    post 29.020 .....

test55 test55
    pre 29.415 .....
    pre 29.416 .....
    event 29.417 .....
    post 29.418 .....
    post 29.419 .....
    post 29.420 .....

test56 test56
    pre 30.015 .....
    pre 30.016 .....
    event 30.017 .....
    post 30.018 .....
    post 30.019 .....
    post 30.020 .....

test57 test57
    pre 30.415 .....
    pre 30.416 .....
    event 30.417 .....
    post 30.418 .....
    post 30.419 .....
    post 30.420 .....

test58 test58
    pre 31.015 .....
    pre 31.016 .....
    event 31.017 .....
    post 31.018 .....
    post 31.019 .....
    post 31.020 .....

test59 test59
    pre 31.415 .....
    pre 31.416 .....
    event 31.417 .....
    post 31.418 .....
    post 31.419 .....
    post 31.420 .....

test60 test60
    pre 32.015 .....
    pre 32.016 .....
    event 32.017 .....
    post 32.018 .....
    post 32.019 .....
    post 32.020 .....

test61 test61
    pre 32.415 .....
    pre 32.416 .....
    event 32.417 .....
    post 32.418 .....
    post 32.419 .....
    post 32.420 .....

test62 test62
    pre 33.015 .....
    pre 33.016 .....
    event 33.017 .....
    post 33.018 .....
    post 33.019 .....
    post 33.020 .....

test63 test63
    pre 33.415 .....
    pre 33.416 .....
    event 33.417 .....
    post 33.418 .....
    post 33.419 .....
    post 33.420 .....

test64 test64
    pre 34.015 .....
    pre 34.016 .....
    event 34.017 .....
    post 34.018 .....
    post 34.019 .....
    post 34.020 .....

test65 test65
    pre 34.415 .....
    pre 34.416 .....
    event 34.417 .....
    post 34.418 .....
    post 34.419 .....
    post 34.420 .....

test66 test66
    pre 35.015 .....
    pre 35.016 .....
    event 35.017 .....
    post 35.018 .....
    post 35.019 .....
    post 35.020 .....

test67 test67
    pre 35.415 .....
    pre 35.416 .....
    event 35.417 .....
    post 35.418 .....
    post 35.419 .....
    post 35.420 .....

test68 test68
    pre 36.015 .....
    pre 36.016 .....
    event 36.017 .....
    post 36.018 .....
    post 36.019 .....
    post 36.020 .....

test69 test69
    pre 36.415 .....
    pre 36.416 .....
    event 36.417 .....
    post 36.418 .....
    post 36.419 .....
    post 36.420 .....

test70 test70
    pre 37.015 .....
    pre 37.016 .....
    event 37.017 .....
    post 37.018 .....
    post 37.019 .....
    post 37.020 .....

test71 test71
    pre 37.415 .....
    pre 37.416 .....
    event 37.417 .....
    post 37.418 .....
    post 37.419 .....
    post 37.420 .....

test72 test72
    pre 38.015 .....
    pre 38.016 .....
    event 38.017 .....
    post 38.018 .....
    post 38.019 .....
    post 38.020 .....

test73 test73
    pre 38.415 .....
    pre 38.416 .....
    event 38.417 .....
    post 38.418 .....
    post 38.419 .....
    post 38.420 .....

test74 test74
    pre 39.015 .....
    pre 39.016 .....
    event 39.017 .....
    post 39.018 .....
    post 39.019 .....
    post 39.020 .....

test75 test75
    pre 39.415 .....
    pre 39.416 .....
    event 39.417 .....
    post 39.418 .....
    post 39.419 .....
    post 39.420 .....

test76 test76
    pre 40.015 .....
    pre 40.016 .....
    event 40.017 .....
    post 40.018 .....
    post 40.019 .....
    post 40.020 .....

test77 test77
    pre 40.415 .....
    pre 40.416 .....
    event 40.417 .....
    post 40.418 .....
    post 40.419 .....
    post 40.420 .....

test78 test78
    pre 41.015 .....
    pre 41.016 .....
    event 41.017 .....
    post 41.018 .....
    post 41.019 .....
    post 41.020 .....

test79 test79
    pre 41.415 .....
    pre 41.416 .....
    event 41.417 .....
    post 41.418 .....
    post 41.419 .....
    post 41.420 .....

test80 test80
    pre 42.015 .....
    pre 42.016 .....
    event 42.017 .....
    post 42.018 .....
    post 42.019 .....
    post 42.020 .....

test81 test81
    pre 42.415 .....
    pre 42.416 .....
    event 42.417 .....
    post 42.418 .....
    post 42.419 .....
    post 42.420 .....

test82 test82
    pre 43.015 .....
    pre 43.016 .....
    event 43.017 .....
    post 43.018 .....
    post 43.019 .....
    post 43.020 .....

test83 test83
    pre 43.415 .....
    pre 43.416 .....
    event 43.417 .....
    post 43.418 .....
    post 43.419 .....
    post 43.420 .....

test84 test84
    pre 44.015 .....
    pre 44.016 .....
    event 44.017 .....
    post 44.018 .....
    post 44.019 .....
    post 44.020 .....

test85 test85
    pre 44.415 .....
    pre 44.416 .....
    event 44.417 .....
    post 44.418 .....
    post 44.419 .....
    post 44.420 .....

test86 test86
    pre 45.015 .....
    pre 45.016 .....
    event 45.017 .....
    post 45.018 .....
    post 45.019 .....
    post 45.020 .....

test87 test87
    pre 45.415 .....
    pre 45.416 .....
    event 45.417 .....
    post 45.418 .....
    post 45.419 .....
    post 45.420 .....

test88 test88
    pre 46.015 .....
    pre 46.016 .....
    event 46.017 .....
    post 46.018 .....
    post 46.019 .....
    post 46.020 .....

test89 test89
    pre 46.415 .....
    pre 46.416 .....
    event 46.417 .....
    post 46.418 .....
    post 46.419 .....
    post 46.420 .....

test90 test90
    pre 47.015 .....
    pre 47.016 .....
    event 47.017 .....
    post 47.018 .....
    post 47.019 .....
    post 47.020 .....

test91 test91
    pre 47.415 .....
    pre 47.416 .....
    event 47.417 .....
    post 47.418 .....
    post 47.419 .....
    post 47.420 .....

test92 test92
    pre 48.015 .....
    pre 48.016 .....
    event 48.017 .....
    post 48.018 .....
    post 48.019 .....
    post 48.020 .....

test93 test93
    pre 48.415 .....
    pre 48.416 .....
    event 48.417 .....
    post 48.418 .....
    post 48.419 .....
    post 48.420 .....

test94 test94
    pre 49.015 .....
    pre 49.016 .....
    event 49.017 .....
    post 49.018 .....
    post 49.019 .....
    post 49.020 .....

test95 test95
    pre 49.415 .....
    pre 49.416 .....
    event 49.417 .....
    post 49.418 .....
    post 49.419 .....
    post 49.420 .....

test96 test96
    pre 50.015 .....
    pre 50.016 .....
    event 50.017 .....
    post 50.018 .....
    post 50.019 .....
    post 50.020 .....

test97 test97
    pre 50.415 .....
    pre 50.416 .....
    event 50.417 .....
    post 50.418 .....
    post 50.419 .....
    post 50.420 .....

test98 test98
    pre 51.015 .....
    pre 51.016 .....
    event 51.017 .....
    post 51.018 .....
    post 51.019 .....
    post 51.020 .....

test99 test99
    pre 51.415 .....
    pre 51.416 .....
    event 51.417 .....
    post 51.418 .....
    post 51.419 .....
    post 51.420 .....

test100 test100
    pre 52.015 .....
    pre 52.016 .....
    event 52.017 .....
    post 52.018 .....
    post 52.019 .....
    post 52.020 .....

test101 test101
    pre 52.415 .....
    pre 52.416 .....
    event 52.417 .....
    post 52.418 .....
    post 52.419 .....
    post 52.420 .....

test102 test102
    pre 53.015 .....
    pre 53.016 .....
    event 53.017 .....
    post 53.018 .....
    post 53.019 .....
    post 53.020 .....

test103 test103
    pre 53.415 .....
    pre 53.416 .....
    event 53.417 .....
    post 53.418 .....
    post 53.419 .....
    post 53.420 .....

test104 test104
    pre 54.015 .....
    pre 54.016 .....
    event 54.017 .....
    post 54.018 .....
    post 54.019 .....
    post 54.020 .....

test105 test105
    pre 54.415 .....
    pre 54.416 .....
    event 54.417 .....
    post 54.418 .....
    post 54.419 .....
    post 54.420 .....

test106 test106
    pre 55.015 .....
    pre 55.016 .....
    event 55.017 .....
    post 55.018 .....
    post 55.019 .....
    post 55.020 .....

test107 test107
    pre 55.415 .....
    pre 55.416 .....
    event 55.417 .....
    post 55.418 .....
    post 55.419 .....
    post 55.420 .....

test108 test108
    pre 56.015 .....
    pre 56.016 .....
    event 56.017 .....
    post 56.018 .....
    post 56.019 .....
    post 56.020 .....

test109 test109
    pre 56.415 .....
    pre 56.416 .....
    event 56.417 .....
    post 56.418 .....
    post 56.419 .....
    post 56.420 .....

test110 test110
    pre 57.015 .....
    pre 57.016 .....
    event 57.017 .....
    post 57.018 .....
    post 57.019 .....
    post 57.020 .....

test111 test111
    pre 57.415 .....
    pre 57.416 .....
    event 57.417 .....
    post 57.418 .....
    post 57.419 .....
    post 57.420 .....

test112 test112
    pre 58.015 .....
    pre 58.016 .....
    event 58.017 .....
    post 58.018 .....
    post 58.019 .....
    post 58.020 .....

test113 test113
    pre 58.415 .....
    pre 58.416 .....
    event 58.417 .....
    post 58.418 .....
    post 58.419 .....
    post 58.420 .....

test114 test114
    pre 59.015 .....
    pre 59.016 .....
    event 59.017 .....
    post 59.018 .....
    post 59.019 .....
    post 59.020 .....

test115 test115
    pre 59.415 .....
    pre 59.416 .....
    event 59.417 .....
    post 59.418 .....
    post 59.419 .....
    post 59.420 .....

test116 test116
    pre 60.015 .....
    pre 60.016 .....
    event 60.017 .....
    post 60.018 .....
    post 60.019 .....
    post 60.020 .....

test117 test117
    pre 60.415 .....
    pre 60.416 .....
    event 60.417 .....
    post 60.418 .....
    post 60.419 .....
    post 60.420 .....

test118 test118
    pre 61.015 .....
    pre 61.016 .....
    event 61.017 .....
    post 61.018 .....
    post 61.019 .....
    post 61.020 .....

test119 test119
    pre 61.415 .....
    pre 61.416 .....
    event 61.417 .....
    post 61.418 .....
    post 61.419 .....
    post 61.420 .....

test120 test120
    pre 62.015 .....
    pre 62.016 .....
    event 62.017 .....
    post 62.018 .....
    post 62.019 .....
    post 62.020 .....

test121 test121
    pre 62.415 .....
    pre 62.416 .....
    event 62.417 .....
    post 62.418 .....
    post 62.419 .....
    post 62.420 .....

test122 test122
    pre 63.015 .....
    pre 63.016 .....
    event 63.017 .....
    post 63.018 .....
    post 63.019 .....
    post 63.020 .....

test123 test123
    pre 63.415 .....
    pre 63.416 .....
    event 63.417 .....
    post 63.418 .....
    post 63.419 .....
    post 63.420 .....

test124 test124
    pre 64.015 .....
    pre 64.016 .....
    event 64.017 .....
    post 64.018 .....
    post 64.019 .....
    post 64.020 .....

test125 test125
    pre 64.415 .....
    pre 64.416 .....
    event 64.417 .....
    post 64.418 .....
    post 64.419 .....
    post 64.420 .....

test126 test126
    pre 65.015 .....
    pre 65.016 .....
    event 65.017 .....
    post 65.018 .....
    post 65.019 .....
    post 65.020 .....

test127 test127
    pre 65.415 .....
    pre 65.416 .....
    event 65.417 .....
    post 65.418 .....
    post 65.419 .....
    post 65.420 .....

test128 test128
    pre 66.015 .....
    pre 66.016 .....
    event 66.017 .....
    post 66.018 .....
    post 66.019 .....
    post 66.020 .....

test129 test129
    pre 66.415 .....
    pre 66.416 .....
    event 66.417 .....
    post 66.418 .....
    post 66.419 .....
    post 66.420 .....

test130 test130
    pre 67.015 .....
    pre 67.016 .....
    event 67.017 .....
    post 67.018 .....
    post 67.019 .....
    post 67.020 .....

test131 test131
    pre 67.415 .....
    pre 67.416 .....
    event 67.417 .....
    post 67.418 .....
    post 67.419 .....
    post 67.420 .....

test132 test132
    pre 68.015 .....
    pre 68.016 .....
    event 68.017 .....
    post 68.018 .....
    post 68.019 .....
    post 68.020 .....

test133 test133
    pre 68.415 .....
    pre 68.416 .....
    event 68.417 .....
    post 68.418 .....
    post 68.419 .....
    post 68.420 .....

test134 test134
    pre 69.015 .....
    pre 69.016 .....
    event 69.017 .....
    post 69.018 .....
    post 69.019 .....
    post 69.020 .....

test135 test135
    pre 69.415 .....
    pre 69.416 .....
    event 69.417 .....
    post 69.418 .....
    post 69.419 .....
    post 69.420 .....

test136 test136
    pre 70.015 .....
    pre 70.
```

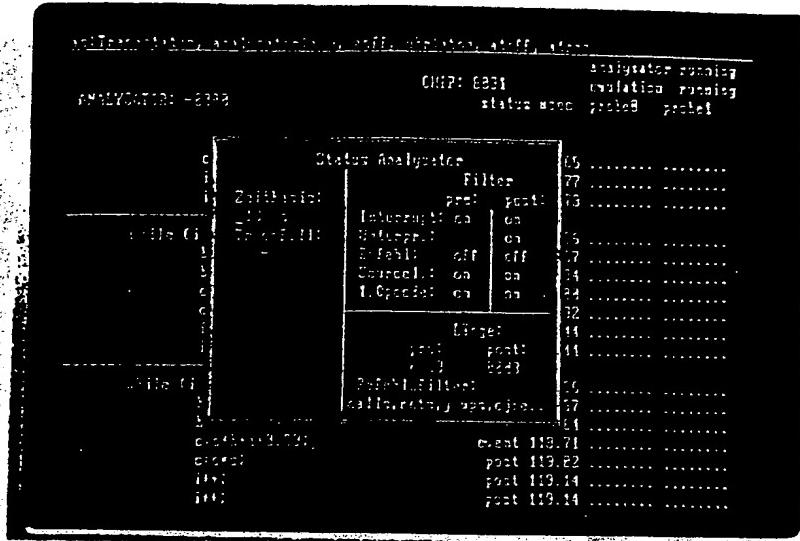


Bild 3. Details zum Analytorestatus

Durch „Filter“ den Trace-Speicher optimal nutzen

Gerade beim Hochsprachen-Trace kann es vorkommen, daß der Trace-Speicher schlecht genutzt wird. Es ist nämlich durchaus keine Seltenheit, wenn eine Hochsprachen-Zeile mehr als 1000 Assemblerbefehle enthält. In einer ungefilterten Aufzeichnung würden so nicht einmal 32 Quellcode-Zeilen in einen 32 KWorte tiefen Trace-Speicher passen!

Im hier beschriebenen Emulator wurden deshalb dem Trace-Speicher verschiedene Filter vorgeschaltet. Durch Setzen dieser Filter kann der Entwickler erreichen, daß nur Daten, die ihn interessieren, aufgezeichnet werden. Dies erhöht einerseits die Übersichtlichkeit der Aufzeichnung, andererseits spart dieses Verfahren Speicherplatz, so daß Langzeittests überhaupt erst möglich werden. Durch Ausnützen der Filter und der sehr leistungsfähigen sequentiellen Breakpoint-Logik kann die Aufzeichnungsdauer extrem verlängert werden (z.B. über Nacht). Dies ist hilfreich bei Langzeittests, wenn also ein Fehler nur selten auftritt. Eine solche Situation wäre z.B. ein Stack-Überlauf nach Aufruf einer Interruptroutine, die ihrerseits eine sehr speicherintensive Hauptprogrammroutine unterbricht.

Folgende Filter, die voneinander unabhängig und für Pre- und Post-Trace aktiviert werden können, wurden implementiert:

- Unterprogramm-Filter: Aufgezeichnet wird nur der Unterprogrammaufruf; das Unterprogramm selbst jedoch nicht. Dadurch kann erreicht werden, daß ein bereits getestetes Unterprogramm nicht mehr aufgezeichnet wird. Dies erhöht die Lesbarkeit einer Trace-Aufzeichnung wesentlich.
- Interrupt-Filter: Das Filter verhindert, daß Interruptanforderungen die Trace-Aufzeichnung des zu testenden Programmteils unterbrechen. Dieses Filter funktioniert im wesentlichen genauso wie das Unterprogramm-Filter.

○ Source-Line-Filter: Dieses Filter wurde speziell für Anwendungen in Hochsprache implementiert. Aufgezeichnet werden nur diejenigen Befehle, die einer Hochsprachenzeile entsprechen. Dies führt zu einer äußerst komprimierten und Trace-Speicher sparenden Aufzeichnung von C-Programmen. Selbstverständlich erfolgt die Anzeige des Trace-Speichers auch auf Source-Code-Ebene.

○ Befehls-Filter: Nur bestimmte Gruppen von Befehlen (z.B. alle Sprungbefehle) werden aufgezeichnet. Dies ist z.B. für folgende Testsituation interessant: Ein Unterprogramm soll getestet werden, wobei wichtig ist, welche Routinen zuvor aufgerufen wurden. Dazu aktiviert man im Pre-Trace das Befehls-Filter und zeichnet nur alle Unterprogrammaufrufe (Befehl „call ..“) auf. Im Post-Trace ist das Filter inaktiv; das gesamte Unterprogramm wird aufgezeichnet.

○ Zyklus-Filter: Die CPUs der Intel-8051-Familie führen pro Befehl bis zu acht Speicherzugriffe durch. Für einen Großteil aller Testanwendungen genügt jedoch der erste Speicherzugriff, da mit diesem der auszuführende Befehl festgelegt wird und die weiteren Opcodes (z.B. RAM-Adressen) später aus dem bekannten Programm rekonstruiert werden können. Ist dieses Filter aktiviert, so werden nur die ersten Speicherzugriffe und alle Zugriffe auf ein externes RAM aufgezeichnet.

○ Frei programmierbares Filter: Spezielle Programmteile können mit einem Cursor markiert und somit zur Aufzeichnung freigegeben werden. Dieses Filter läßt sich sehr flexibel einsetzen (siehe auch Bild 3).

Stefan Richt studierte Elektrotechnik an der TU München und ist seit 1986 bei der Firma Richt-Magneton GmbH in der Entwicklung tätig. Nebenberuflich betreibt er ein Entwicklungslabor, in dem u.a. der vorgestellte In-Circuit-Emulator entwickelt wurde.

Peter Hamm studierte Physik an der TU München und arbeitete freiberuflich seit 1987 bei der Firma Richt-Magneton GmbH. Bei der Entwicklung des vorgestellten In-Circuit-Emulators war er maßgeblich beteiligt.